Remarks/Arguments

Claims 1-42 are pending in the application. Claims 1, 21 and 42 are independent.

The Examiner has rejected claims 11 and 31 under 35 U.S.C. 112 for being indefinite because it is not clearly understood what "ECMA" means. However, both claims 11 and 31 refer to ECMAscript which will be understood by a person of ordinary skill in the art as a scripting language, standardized by Ecma International in the ECMA-262 specification. Accordingly, it will be appreciated that ECMAscript is the name of the script and not an abbreviation.

The Examiner has further rejected claims 21-41 under 35 U.S.C. 101 for being directed to a system that does not comprise the necessary hardware. Accordingly, the claim have been amended so that the system includes a memory for storing instructions and a computer processor for implementing the instructions to provide the functionality as claimed. Support for the amendments can be found in the Specification from page 8, line 1 to page 9, line 20.

The Examiner has further rejected claims 1-10, 12-30, 32-42 under 35 U.S.C. 102(e) as being anticipated by Sharma (US 2003/0204645). Applicant respectfully traverses the rejections.

Claim 1 recites a method for providing dynamic interaction between a pair of application programs by an interface module of a terminal, the pair of applications including a requestor application desiring access to a target application, the method comprising the steps of:

registering access information of the target application, the access information including published access information made available in a data structure for retrieval by the interface module;

receiving an access request by the interface module from the requestor application, the access request including request content corresponding to the published access information of the target application;

obtaining an interface component by using the request content to search the data structure, the interface component configured for enabling communication between the interface module and the target application in an access format expected by the target application; and

employing the interface component by the interface module to satisfy the access request of the requestor application for interaction with the target application.

As described in the Background, the problem the subject invention addresses relates to communication between related applications. Specifically, if an application interface changes, it is also required to change many, if not all, of the related or dependent applications to maintain compatibility.

Effectively, the solution as claimed provides an interface module that facilitates the communication between a requestor application and a target application. The target application registers access information, such as an application name and its corresponding parameters. The requestor application submits an access request for the target application, which is received by the interface module. The requestor application also provides content corresponding to the published access information of the target application.

The interface module uses the received information (in a generic format such as XML, for example) to obtain an interface component, which is configured to enable communication between the platform neutral interface and an access format expected by the target application. The interface component is then used by the platform neutral interface to satisfy the access request.

Therefore, it can be seen that the platform neutral interface acts as an intermediary agent on behalf of the requestor application to facilitate communication. In this way, requestor applications only have to know how to communicate with the interface module. The platform neutral interface can then communicate with the target application via the interface component. Accordingly, if the target application changes, all that needs to be changed is the interface component and, possibly, the published data.

In contrast, Sharma explicitly teaches communicating directly between two endpoints. As described in the Summary of the Invention, Sharma teaches a computing system providing a service associated with a service endpoint class that implements a service endpoint interface. The interface is facilitated by packaging the service endpoint class and interface in an archive file and deploying it on a servlet container.

Further, the computing system can create a WSDL document that the describes the service endpoint interface. The WSDL document can then be used by a client to access the service endpoint. As described by Sharma in paragraphs 111 to 115, in order to access the service, the client retrieves the WSDL document and employs a WSDL-to-Java mapping tool to generate client artifacts from the WSDL document.

Once the artifacts have been generated, the client can use them to invoke a remote method at the service endpoint.

Thus it will be apparent that, unlike the present invention, Sharma teaches a system in which there is direct point-to-point communication between an interface at a client and an interface at a service endpoint.

In view of the differences described above, there are a number of elements of claim 1 that are not present in Sharma.

For example, the interface module (formerly referred to as the platform neutral interface) is equated by the Examiner with the server side API 115. However, as described in paragraph 43 or Sharma, "Server side API 115 may be one or more programming interfaces that enable server side JAX-RPC runtime system 114 to communicate with other software operating in server 110." Accordingly, there is no teaching of the server side API retrieving published access information of a target application. Nor is their any teaching of the server side API receiving an access request from the client including request content corresponding to published access information .

Further, although the Examiner appears to have distinguished between server side APIs (equating them to the interface module) and server side JAX-RPC APIs (equating them to the interface component), Sharma reaches in paragraph 43 that the server side APIs are server side JAX-RPC APIs. Thus, in accordance with Sharma, the two APIs are one and the same and, therefore, cannot represent both the interface module and the interface component, as suggested by the Examiner.

Accordingly, even if it could be argued that Sharma teaches an interface module, then it cannot be argued that Sharma teaches an interface component. Similarly, if it could be argued that Sharma teaches an interface component, then it cannot be argued that Sharma teaches an interface module. Since Sharma teaches a point-to-point communication between a client and a service endpoint, there is no need for both an interface module and an interface component.

Therefore, for at least the reasons discussed above, Applicant submits claim 1 is patentable in view of Sharma and, as such, requests that the rejection of claim 1 be withdrawn.

Independent claims 21 and 42 are similar in scope to claim 1, and therefore a similar argument applies. Accordingly, we submit that the rejection to these claims be withdrawn for at least the same reasons.

Since the remaining dependent claims depend from one of the above noted independent claims, since we submit that the rejection of these claims be withdrawn for at least the same reasons.

For the foregoing reasons, the Applicant respectfully submits that the claimed invention is patentable over the prior art. Reconsideration and allowance of the claims is respectfully requested.

Respectfully submitted,

/Jonathan Pollack/

Jonathan Pollack
Registration No. 49,065
416 862 5405

Gowling Lafleur Henderson LLP
1 First Canadian Place, Suite 1600
Toronto, Ontario, M5X 1G5

TOR_LAW\ 7040787\1